COOLEY GODWARD LLP
ATTORNEY DOCKET NO.: CNTW-011/00US
CLIENT NO.: 036958-2014

APPLICATION FOR PATENT

TITLE: **SYSTEM AND METHOD FOR GENERATING A REPRESENTATION OF A CONFIGURATION SCHEMA**

INVENTOR(S): **MIKE COURTNEY**

RELATED APPLICATIONS

[0001] The present application is related to commonly owned and assigned application Nos.:

09/730,864, entitled *System and Method for Configuration, Management and Monitoring of Network Resources*, filed December 6, 2000;

09/730,680, entitled *System and Method for Redirecting Data Generated by Network Devices*, filed December 6, 2000;

09/730,863, entitled *Event Manger for Network Operating System*, filed December 6, 2000;

09/730,671, entitled *Dynamic Configuration of Network Devices to Enable Data Transfers*, filed December 6, 2000;

09/730,682, entitled *Network Operating System Data Directory*, filed December 6, 2000;

09/799,579, entitled *Global GUI Interface for Network OS*, filed March 6, 2001;

CNTW-007, entitled *System and Method for Generating a Configuration Schema,* filed August 29, 2001; and

CNTW-008, entitled *System and Method for Modeling a Network Device's Configuration,* filed August 29, 2001.

all of which are incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to network device interrogation and configuration. In particular, but not by way of limitation, the present invention relates to systems and methods for interrogating and configuring routers, switches, hubs, and/or optical components.

BACKGROUND OF THE INVENTION

[0003] Networks, and in particular, the Internet, have revolutionized communications. Data vital to the continued prosperity of the world economy is constantly being exchanged between end-users over these networks. Unfortunately, the expansion and maintenance of these networks is outpaced by the demand for additional bandwidth. Network equipment is often difficult to configure, and qualified network technicians are in extremely short supply. Thus, many needed network expansions and upgrades must be delayed until these technicians are available. While these upgrades and expansions are pending, end-users continue to suffer poor network performance.

[0004] For example, Cisco™ routers are notoriously difficult to configure--especially in light of the new XML-based interfaces introduced by competitors such as Juniper Networks™. Instead of a user-friendly XML-based interface, Cisco uses a cumbersome command line interface (CLI) for its routers. Cisco's CLI interface is the result of many years of semi-controlled modifications to its router operating systems and has resulted in a tangled mess of commands and subcommands.

[0005] If Cisco attempted to abandon its CLI in favor of the new user-friendly XML-based interface, many years of development and expertise could be lost. Moreover, even if it could develop an XML-based interface, there is presently no economical way to integrate it into the thousands of existing routers. Despite the difficulties in implementing a more user-friendly interface, to remain competitive, Cisco and similarly situated companies need to move away from the CLI. However, present technology does not provide these companies with an acceptable option that allows continued use of its extensive CLI knowledge base while simultaneously providing system administrators with a user-friendly interface, e.g., XML-based interface. Moreover, present technologies do not provide an acceptable way to provide backward compatibility with existing devices.

[0006] Cisco is not the only manufacturer to face this interface-upgrade problem. Many manufacturers would like to continue using their existing interface knowledge base while providing system administrators a friendly, consistent interface. Accordingly, a system and method are needed that will allow manufacturers, like Cisco, to create user-friendly interfaces for both next-generation and existing devices.

SUMMARY OF THE INVENTION

[0007] Exemplary embodiments of the present invention that are shown in the drawings are summarized below. These and other embodiments are more fully described in the Detailed Description section. It is to be understood, however, that there is no intention to limit the invention to the forms described in this Summary of the Invention or in the Detailed Description. One skilled in the art can recognize that there are numerous modifications, equivalents and alternative constructions that fall within the spirit and scope of the invention as expressed in the claims.

[0008] Embodiments of the present invention can provide a system and method for generating a configuration schema and/or a representation thereof for network devices. Other embodiments can provide a system and method for configuring network devices using a configuration schema and/or a representation of the schema. These and other embodiments are discussed more fully below.

[0009] Although schema are generally used to validate commands, in one implementation of the present invention, the configuration schema can be used to generate commands. For example, a configuration command can be retrieved from a Cisco router. This configuration command is generally expressed in terms of a CLI-based command structure. Using the XML configuration schema, however, the CLI-based commands can be converted to an XML format, which is significantly more manageable than a CLI-based format. Once the CLI-based command has been converted

to an XML format, the XML version of the command can be easily passed between various computers and system administrators in a highly readable, standardized format.

[0010] In another implementation, the schema can be used to generate CLI commands from, for example, XML-based commands. As previously described, the configuration schema contains the commands, command hierarchy and bounds of the various configuration commands. When given a command in XML format, the command information in the configuration schema can be used to reformat the XML-based command into a proper CLI format. Once reformatted into a CLI format, the command can be pushed out to the appropriate router. Thus, a system administrator could configure such a router without knowing the specifics of the CLI.

[0011] In another embodiment of the present invention, a representation of the schema is substituted for the schema or used in conjunction with the schema. This representation can be used anywhere that the schema can be used, but can provide significant "runtime" advantages versus the original schema. For example, the representation of the schema can be used to validate commands, generate commands, drive a graphical user interface, etc.

[0012] The representation of the schema can be generated by traversing the schema and creating, for example, a hash table or hash map with entries linked to corresponding container objects. These container objects can include information from the schema and

other data, e.g., metadata, which may not be included in the schema, such as device manufacturer, device type, device model, device operating system, etc.

[0013] Advantages of the representation of the schema can include a smaller memory footprint than the complete schema, decreased traversal time when generating or converting commands, and ease-of-use relative to the original schema. As previously stated, however, the above-described embodiments, implementations, and advantages are for illustration purposes only. Numerous other embodiments, implementations, and details of the invention are easily recognized by those of skill in the art from the following descriptions and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Various objects and advantages and a more complete understanding of the present invention are apparent and more readily appreciated by reference to the following Detailed Description and to the appended claims when taken in conjunction with the accompanying Drawings wherein:

FIGURE 1 is a block diagram of a conventional network;

FIGURE 2 is a block diagram of a conventional router;

FIGURE 3 is a flowchart of a method for generating a configuration schema in accordance with one embodiment of the present invention;

FIGURE 4 is a representation of one storage model for storing configuration schema across different device types, manufacturers, models and operating system versions;

FIGURE 5 is a block diagram of a router constructed in accordance with one embodiment of the present invention;

FIGURE 6 is a block diagram of one embodiment of the present invention;

FIGURE 7 is a block diagram of another embodiment of the present invention;

FIGURE 8 is a flowchart of one method for configuring a router using a configuration schema;

FIGURE 9 is a flowchart of one method for generating a representation of a configuration schema; and

FIGURE 10 is a block diagram of one embodiment of the present invention incorporating a representation of the configuration schema.

DETAILED DESCRIPTION

[0015] Referring now to the drawings, where like or similar elements are designated with identical reference numerals throughout the several views, and referring in particular to FIGURE 1, it illustrates a block diagram of a conventional network system 100. In this network system 100, end-users 105 are connected to servers 110, which are connected to networking equipment such as hubs, (not shown) optical components 115, and routers 120. Using the networking equipment, end-users 105 associated with different servers 110 can exchange data.

[0016] As new servers 110 and end-users 105 are added to the overall system 100, or as new software becomes available, the routers 120 and/or optical components 115 of the network system 100 may need reconfiguring. To reconfigure these components, a system administrator 125--with the proper authorization--could access the router 120 and/or optical component 115 by, for example, establishing a telnet connection to the component and transferring configuration instructions thereto.

[0017] Referring now to FIGURE 2, it is a block diagram of a conventional router. In this representation, a processor 125 is connected to a configuration interface 130, an operating system (OS) storage module 135, a command storage module 140, a configuration storage module 145, and a routing module 150. The illustrated arrangement of these components is logical and not meant to be an actual hardware diagram. Thus, the components can be combined or further separated in an actual implementation. Moreover, the construction of each individual component is well-known to those of skill in the art.

[0018] When a system administrator 125 wishes to reconfigure a router 120, he accesses the router 120 through the configuration interface 130 and retrieves the present configuration for the router 120 from the configuration storage module 145. The system administrator 125 can review available configuration commands and bounds--usually in a CLI format--by accessing and reviewing the commands stored in the command storage module 140. In essence, the command storage module 140 provides the knowledge base

for a "help" screen. The commands stored in the command storage module 140 are generally unique to the particular OS version stored in the OS module 135.

[0019] After the system administrator 125 has constructed the new configuration instructions, these instructions are pushed through the configuration interface 130 and stored in the configuration storage module 145. For Cisco routers, interaction is generally through a CLI. In other words, the command storage module 140 is queried through the CLI; available commands are returned through the CLI; and new configuration commands are provided to the router 120 through the CLI. Unfortunately, the CLI is difficult to manage and requires highly skilled technicians for even simple tasks.

[0020] Referring now to FIGURE 3, it is a flowchart of one method for generating a configuration schema in accordance with the principles of the present invention. The illustrated method can be used, for example, to generate an XML schema from the CLI commands associated with a Cisco router. In accordance with the principles of the present invention, one method for constructing a configuration schema involves a system administrator 125 (in conjunction with an automated system) connecting to a router 120 through, for example, a telnet connection. Next, the system administrator 125 logs into the router 120 and activates a command extraction mode (steps 160 and 165). With regard to a Cisco router, the command extraction mode is activated by entering a "?" at the prompt. Next, the system administrator 125 retrieves the primary commands, subcommands and bounds (steps 170, 175 and 180). This retrieval can be done through

an automated, recursive search. For a Cisco router, the following search could be executed and the following results returned where ">" is the CLI prompt:

```
> ?
router
admin
global
> router?
bgp
ospf
>
```

This process could be repeated until termination for each command and subcommand.

[0021] Once the commands, subcommands, and bounds are collected, they can then be recorded and cleansed (steps 185 and 190). Duplicate commands, for example, could be identified. When these duplicate commands include different subcommands and/or bounds, a single, cleansed command can be constructed to replace the duplicate commands. The cleansed commands, assuming that cleansing was necessary, can then be used to build a configuration schema, which in essence is a modeling of the router's command structure (step 195). An example snippet of such a modeling in an XML schema is represented by:

```
<xsd:element name="vlan">
    <xsd:complexType>
        <xsd:choice>
            <xsd:sequence>
                <xsd:element name="mapping">
            <xsd:complexType/>
        </xsd:element>
        <xsd:element name="dot1q"  fixed="dot1q">
            <xsd:complexType/>
```

```
                <xsd:element>
                <xsd:element name="ARG.001".
                        <xsd:simpleType>
                                .
                                .
                                .
                </xsd:choice>
        </xsd:complexType>
</xsd:element>
```

[0022] In one embodiment, the conversion between the text file and the XML configuration schema is performed by a Visual Basic program. This program identifies arrays of related commands in the text file. Individual arrays can be identified, for example, because they are generally separated by termination characters or by logical termination indicators. Additionally, when an input indicator is encountered in the text file, the program can insert a placeholder into the configuration schema to represent the input indicator. This placeholder can then be associated with the bounds for that particular input. For example, if the input corresponding to the input indicator should be between 1 and 10, a bound of 1 to 10 can be associated with the placeholder.

[0023] After the configuration schema has been generated, it is associated with characteristics of the router and stored accordingly (steps 200 and 205). For example, the configuration schema might be associated with a Cisco router, model 7500, OS version 12.0. A representation of a storage model 210 for storing configuration schema according to manufacturer, device type, device model, and OS version is shown in Figure 4. The first data block 215 is dedicated to Cisco routers as indicated in the upper left-hand corner. Each row represents a different model of Cisco device, and each column

represents a different OS version. Similarly, the second data block is for Juniper™ routers 220 and the third is for Ciena™ devices 225.

[0024] Referring now to FIGURE 5, it is a block diagram of a router 230 constructed in accordance with one embodiment of the present invention. In this embodiment, a converter 235 and schema storage module 240 are added to the router of FIGURE 2. The router 230 is generally configured to interface with the system administrator 125 through the configuration interface. Even though the router 230 operates through a CLI configuration interface, a system administrator 125 can reconfigure such a router using XML-based commands--assuming the configuration schema stored in the schema storage module 240 is an XML schema. For example, a system administrator 125 can send an XML-based command to the configuration interface 130. That XML-based command can be passed to the converter 235 which converts the XML-based command to a CLI-based command using the XML schema. A CLI-based command, not the XML command, can then be passed to the configuration storage module 145 where it is integrated into the configuration of the router.

[0025] Referring now to FIGURE 6, it is a block diagram 245 of an embodiment of the present invention in which the converter and schema storage 245 are localized within the system administrator 125. Rather than components within the router 120 converting an XML-based command to a CLI-based command, the localized converter 235' and schema storage module 240' convert the XML-based command to a CLI-based command and then transmit the CLI-based command through the network 250 to the router 120.

[0026] Referring now to FIGURE 7, it shows a block diagram 255 of yet another embodiment of the present invention. In this embodiment, the converter 235''and schema storage module 240'' are distributed relative to both the router 120 and the system administrator 125. Thus, any XML-based configuration commands generated by the system administrator 125 are sent through the network 250 to the converter 235.'' Using the configuration schema stored in the schema storage module 240'', the converter 235'' can convert the XML-based command to a CLI-based command and send that command through the network 250 to the router 120.

[0027] FIGURE 8 illustrates one method of operating the system of FIGURE 7. In this method, the converter 235'' initially receives an XML-based configuration command (step 260). The converter 235'' then determines the manufacturer, model and/or OS version for the router 120 to which the command is directed and accesses the appropriate configuration schema for that router 120. The (steps 265 and 270) converter 235'' then generates a CLI-based command from the XML-based command, verifies the correctness and validity of that command, and provides it to the router 120 (steps 275 and 280).

[0028] Referring now to FIGURE 9, it is a flowchart of one method for generating a representation of a configuration schema. In this embodiment, a representation is generated for each schema. For example, if a first schema was associated with a particular class of Cisco routers and a second schema was associated with a particular class of Juniper routers, then each schema could have a corresponding, unique representation.

[0029] To generate a representation of a particular configuration schema, one embodiment of the present invention retrieves a command from the previously assembled configuration schema (step 290). Additionally, any related commands, e.g., parent commands, child commands, sibling commands, root commands, etc., in the configuration schema can be identified and retrieved. The retrieved command and the retrieved parent commands can then be used to generate a unique hash key for the retrieved command (step 295). An example of such a hash key is:

configuration | interface | FastEthernet.

This type of key can be generated by concatenating indicators of different commands. For example, a hash key can be created by concatenating a parent command, such as "configuration," with a sibling command, such as "interface."

[0030] After the unique hash key is generated, a corresponding hash object can also be generated. This hash object can include basic information related to the generated hash key. To generate the schema hash object, information such as data type, sibling commands, and application specific information can be retrieved and assembled into the schema object (steps 300, 305, 310 and 315). The data type information, for example, can indicate whether the data associated with a particular command is a string, an integer, etc. and the sibling information can identify commands at the same hierarchical level as the initially retrieved command that have the same parent command as the initially retrieved command. Additionally, in certain embodiments, specialized application information can also be retrieved from the schema or another source. This application

information, for example, can define special processing requirements for a schema. Certain embodiments of the schema hash object also include metadata such as device name, device type, device manufacturer, operating version, etc.

[0031] An exemplary hash object is shown in Example 1. The hash object in Example 1 is associated with the key "configuration | interface | FastEthernet."

EXAMPLE 1: Hash Object

```
name=FastEthernet
parent=interface
hierarchy=configuraiton|interface
type=null
container=choice
isContianterOptional=false
containerlist=[choice]
childContainer=complextype
isBounded=false
isOptional=false
isBoolean=false
childElements=[ARG.001, no, arp, random-detect, bandwidth, cdp,
        description, ip, encapsulaiton, fair-queue, hold-queue, keepalive,
        logging, priority-group, standby, shutdown]
siblingElements=[]
documentation=Fast Ethernet Interface
appInfo=[VALUE_CONCATENATED]
isUseRestOfLine=false
isValueConcatenated=true
restriction base=null
minLength=null
minInclusive=null
maxInclusive=null
```

[0032] This use of key-based objects has several significant advantages. First, because each schema node of interest is represented in the hash by a key and an associated object, getting to an object via the key is very quick compared to having to traverse through the original schema. Since a direct lookup is possible with the key approach, this

significantly reduces the runtime processing required to get schema node information. Second, the object generally contains only the information of interest about the schema node. Thus, extraneous node layers and node data do not have to be decoded and then ignored during the runtime processing. This also reduces processing time. Third, the fact that the schema nodes are manifested as generic programming objects provides ease-of-use for "consumers" of the schema. These consumers, which are other processes, do not have to have built-in redundant knowledge of how to properly traverse and decode the schema objects. This also makes debugging and maintenance much easier since this more difficult code only exists in one place. Fourth, a schema node object can be retrieved from the hash and passed to other programs without having to pass the entire schema or a disjointed part thereof.

[0033] Hash objects can be divided into groups of hash objects, referred to as hashlets. Hashlets can be organized in a variety of ways. For example, hashlets can be formed by grouping hash objects related to a certain level of configuration commands, such as all sibling commands to the "interface" command. In other embodiments, hashlets can be formed by grouping hash objects related to certain tasks. For example, a hashlet can be formed of hash objects related to Quality of Service (QOS) commands.

[0034] Once a hash object has been assembled, the unique key can be stored in a hash map in association with the hash object. If there are any more commands in the schema that need to be modeled (decision 325), branch 330 is followed and the next command can be retrieved (step 290). If all of the commands have been modeled, then branch 335

can be followed and the various hash objects can be stored as a completed hash table (step 340).

[0035] Referring now to FIGURE 10, it is a block diagram of one embodiment of the present invention incorporating a representation of the configuration schema. In this embodiment, a schema hash generator 345, which is one implementation of a system for generating a representation, is connected with a schema storage module 240, a schema hash key storage module 350, and a schema hash object storage 355. The schema hash storage module 350 and the schema hash object storage module 355 can be a single memory device or multiple memory devices.

[0036] Data from the schema hash key storage module 350 and/or the schema hash object storage module 355 can be used in various network management applications. For example, the hash keys and the hash objects can be used by a device-neutral command generator, such as an XML command generator 360, to generate device-neutral commands from device-native commands. In other embodiments, the hash keys and the hash objects can be used by a device-native command generator, such as a CLI command generator 365, to generate device-native commands from device-neutral commands. In yet another embodiment, the hash keys and the hash objects can be used by a graphical user interface (GUI) to provide a system administrator with a simplified, device-neutral interface for managing a series of network devices.

[0037] Referring now to FIGURE 11, it is a block diagram of one implementation of a representation of the configuration schema. In this implementation, a schema hash key storage module 350 and a schema hash object storage module 355 are connected to a configuration manager 360. The configuration manager 360 can use data from the hash key storage module 250 and the schema hash object storage module 355 to interface with the network device 365.

[0038] Although the embodiments of the present invention are generally described with regard to a router and in particular with regard to Cisco routers, one skilled in the art can understand that the embodiments of the present invention can incorporate routers from most router manufacturers and many other types of network components such as hubs, switches, and optical components. Thus, those skilled in the art can readily recognize that numerous variations and substitutions may be made in the invention, its use and its configuration to achieve substantially the same results as achieved by the embodiments described herein. Accordingly, there is no intention to limit the invention to the disclosed exemplary forms. Many variations, modifications and alternative constructions fall within the scope and spirit of the disclosed invention as expressed in the claims.